EMPLOYEE.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace EmployeeValidation

{

public class Employee

{

/\*

\* Do not modify the return types of the below properties

\*

\*/

public string EmployeeId { get; set; }

public string EmployeeName { get; set; }

public string EmailId { get; set; }

public string DateOfJoining { get; set; }

}

// Do not add new constructors

}

EMPLOYEEVALIDATOR.CS

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Data;

using System.Collections;

using System.Xml.Linq;

using System.Linq;

using System.IO;

using System.Text.RegularExpressions;

namespace EmployeeValidation

{

public class EmployeeValidator

{

/\*

\* Do not remove the attached TestProject. It is meant for auto evaluation of your source code.

\* Do not attach any test classess to the attached test project.

\* Do not attach any new test projects.

\* You are not required to write any automated test cases. You are supposed to write only the code.

\*/

public void ProcessData(string xmlFilePath, string xmlFileName, SqlConnection connection)

{

//Do your logic here

List<Employee> allEmployees = new List<Employee>();

List<Employee> validEmployees = new List<Employee>();

//Step 1

//ReadAllEmployeesFromXmlFile

allEmployees = ReadAllEmployeesFromXmlFile(xmlFilePath, xmlFileName);

//Step 2

//PickValidEmployees

validEmployees = PickValidEmployees(allEmployees);

//Step 3

//SaveValidEmployeesToDataBase

SaveValidEmployeesToDB(validEmployees, connection);

}

public List<Employee> ReadAllEmployeesFromXmlFile(string xmlFilePath, string xmlFileName)

{

//Read the employee details from the xml file and return it in List collection

//Do not hardcode the filename and the file path here

List<Employee> empList = new List<Employee>();

if (File.Exists(xmlFilePath + xmlFileName))

{

//XElement element = XElement.Load(xmlFilePath + xmlFileName);

//IEnumerable<XElement> allEmployees = element.Elements();

//foreach (var Employee in allEmployees)

//{

// Employee emp = new Employee();

// emp.EmployeeId = Employee.Element("EmployeeId").Value;

// emp.EmployeeName = Employee.Element("EmployeeName").Value;

// emp.EmailId = Employee.Element("EmailId").Value;

// //Do not return the date with time appended to it.

// emp.DateOfJoining = Convert.ToDateTime(Employee.Element("DateOfJoining").Value).ToString("MM/dd/yyyy");

// empList.Add(emp);

//}

StreamReader str = new StreamReader(xmlFilePath + xmlFileName);

string txtData = str.ReadToEnd().Replace("\r\n", "");

string[] a = txtData.Split(';');

for (int i = 0; i < a.Length; i++)

{

if (a[i] != "")

{

string[] b = a[i].Split(',');

Employee emp = new Employee();

emp.EmployeeId = b[0];

emp.EmployeeName = b[1];

emp.EmailId = b[2];

emp.DateOfJoining = b[3];

empList.Add(emp);

}

}

return empList;

}

else

{ throw (new EmployeeValidatorException("File Not Found")); }

}

public List<Employee> PickValidEmployees(List<Employee> employees)

{

List<Employee> validEmpListtmp = new List<Employee>();

List<Employee> validEmpList = new List<Employee>();

List<Employee> invalidEmpList = new List<Employee>();

//Pick the valid employees from the List collection

//Return the valid employees in a List

Regex isNumeric = new Regex(@"^\d+$");

Regex isAlphanumeric = new Regex(@"\w+$");

//Regex isValidEmail = new Regex(@"^(?("")(""[^""]+?""@)|(([0-9a-z]((\.(?!\.))|[-!#\$%&'\\*\+/=\?\^`\{\}\|~\w])\*)(?<=[0-9a-z])@))(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]\*[0-9a-z]\*\.)+[a-z0-9]{2,17}))$");

Regex isValidEmail = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$", RegexOptions.IgnoreCase);

Regex isValidDate = new Regex(@"(0[0-9]|1[0-2])\/(0[0-9]|1[0-9]|2[0-9]|3[0-1])\/(19|20)\d{2}$");

validEmpListtmp = employees.GroupBy(p => p.EmployeeId).Select(g => g.First()).ToList();

foreach(var employee in validEmpListtmp)

{

if(isNumeric.IsMatch(employee.EmployeeId) && isAlphanumeric.IsMatch(employee.EmployeeName) && isValidEmail.IsMatch(employee.EmailId) && isValidDate.IsMatch(employee.DateOfJoining))

{

validEmpList.Add(employee);

}

else

{

invalidEmpList.Add(employee);

}

}

FileStream writer = new FileStream(@"C: \Users\282157\Desktop\DotNet\_CSharp\_VB\_Completed\EmployeeValidator\_QB\Input File\ErrorLog.txt", FileMode.Append);

StreamWriter swtr = new StreamWriter(writer);

foreach (Employee emp in invalidEmpList)

{

string text = emp.EmployeeId + "," + emp.EmployeeName + "," + emp.EmailId + "," + emp.DateOfJoining + ";";

swtr.WriteLine(text);

}

swtr.Flush();

swtr.Close();

writer.Close();

return validEmpList;//Return only valid employees in List

}

public void SaveValidEmployeesToDB(List<Employee> employees, SqlConnection connection)

{

//Do not Prefix Database name in the SQL Query. Query should be "Insert into SBA.TableName"

//Should not be "Insert into DatabaseName.SBA.TableName"

//Do not hardcode the connection string here

string query = "INSERT INTO SBA.Employees(EmployeeId, EmployeeName, DateOfJoining, EmailId) VALUES(@empID, @empName, @EmailID, @DOJ)";

connection.Open();

using (SqlCommand cmd = new SqlCommand(query, connection))

{

foreach (Employee emp in employees)

{

cmd.Parameters.AddWithValue("@empID", emp.EmployeeId);

cmd.Parameters.AddWithValue("@empName", emp.EmployeeName);

cmd.Parameters.AddWithValue("@EmailID", emp.EmailId);

cmd.Parameters.AddWithValue("@DOJ", emp.DateOfJoining);

int i = cmd.ExecuteNonQuery();

cmd.Parameters.Clear();

}

}

connection.Close();

}

}

}

EMPLOYEEVALIDATOREXCEPTION.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace EmployeeValidation

{

public class EmployeeValidatorException : Exception

{

public EmployeeValidatorException()

: base()

{

}

public EmployeeValidatorException(string message)

: base(message)

{

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Text;

namespace EmployeeValidation

{

public class Program

{

public static void Main()

{

/\*

\* Pass the file path, file names and connection string if any in this method alone.

\* Do not hardcode in any other methods

\*/

SqlConnection connection = new SqlConnection(@"Data Source=PCName\\SQLEXPRESS;Initial Catalog=DBEmployeeValidation;Integrated Security=True");

EmployeeValidator empValidator = new EmployeeValidator();

try

{

empValidator.ProcessData(@"C:\Users\282157\Desktop\DotNet\_CSharp\_VB\_Completed\EmployeeValidator\_QB\Input File\", "InputFile.txt", connection);

}

catch (EmployeeValidatorException ex)

{

Console.WriteLine(ex.Message.ToString());

Console.ReadLine();

}

}

}

}